Network Working Group
Request for Comments #178
NIC 7118

Ira W. Cotton
June 27, 1971
The MITRE Corporation

7118
Postel

NETWORK GRAPHIC ATTENTION HANDLING

1.0 INTRODUCTION

Discussions of network graphic protocols have thus far primarily dealt with protocols for the description of graphic data to be displayed. RFC 86 proposed a Network Standard Graphic Data Stream (NGDS) which would serve to convey graphic images expressed in the Network Standard Display List (NGDL). RFC 94 expanded on this proposal, and pointed out some shortcomings of the original scheme. RFC 125 also replied to RFC 86 with comments and extensions, but also recognized that a protocol for graphic display alone is insufficient to support an interactive graphic system.

1.1 Topics Covered

The present paper addresses itself to this requirement. The process of attention handling is briefly described, various graphic configurations are discussed, input devices are surveyed to identify the types of data which they produce, and an attention protocol is suggested.

1.2 Viewpoint

It should be made clear at the onset that the discussions which follow will be from the viewpoint of a graphics user or a graphic application program serving one or more users. Our concern is with third-level protocols only. We assume the network is capable of delivering arbitrary bit streams from terminal to graphic application program, but don't care how this is accomplished.

2.0 ATTENTION-HANDLING

In order to demonstrate the need for an attention protocol, we must first define what is meant by "attention" and "attention-handling." We therefore begin by borrowing the definitions given in a recent survey of this area(1).

2.1 Definition

Graphic attention handling refers to the processes and techniques whereby human inputs to a computer graphic system are serviced. An attention event, or simply "attention," is a stimulus to the graphic system, such as that resulting from a keystroke or light pen usage, which presents information to the system. Servicing includes accepting or detecting the hardware input, processing it to determine its intended meaning, and either passing this information to a user routine or taking some immediate action related to the display and/or its underlying data structure, or both. The emphasis is on "immediate." Attention-handling is not intended to include any detailed, application-oriented processing which the attention information may indicate is to be performed. Thus, attention handling may be considered separately from any particular application.

2.2 Independent From Display Considerations

Not only may attention handling be considered separately from any application, but attention generating hardware may be considered separately from display hardware. Oftentimes, it is only coincidental that they come in the same package. Indeed, in some configurations an

input be processed locally (by the terminal) to provide the appropriate response. For example, a keystroke may or may not cause a character to be displayed on a terminal, and the logic causing the display may or may not be local (within the terminal). The keystroke might be immediately displayed locally, as in the case of an alphanumeric display terminal which buffers keystrokes and transmits messages of many characters; or it might be transmitted to the host computer and "echoed" back for display as in teletype-like terminals.

The question is not limited to such simple input devices as keyboards. So-called "intelligent terminals" with integrated programmable logic or even complete small computers can process more sophisticated attentions locally, and even alter a local distillate of the central (host) data structure without central knowledge. This raises the problem of insuring that the display and the graphic application program do not get "out of sync," and requires a more expressive protocol from terminal to host processor.

3.0 SYSTEM CONFIGURATIONS

We now turn to a consideration of the evolution of system configurations for computer graphics. Our intent is to demonstrate that just as display generation has evolved from the output of device dependent codes to a generalized protocol, so too should attention generation evolve.

3.1 Stand-Alone Configuration

Figure 1 illustrates the stand-alone graphic configuration which was the first and is still the most common. As we have stressed, input and output are entirely independent, and are shown as separate devices. In this configuration, display code generation and interrupt processing are both done within the graphic application program in the host processor. The graphic application program is very device-dependent.

3.2 Stand-Alone Configuration with Standardized Formats

The significant conceptual change occurs when the input and output processors are removed from the graphic application program. The graphic application program then generates output and accepts input in a generalized form, as illustrated in Figure 2. The important fact to note is that in order to accommodate additional (different) input and/or output devices, only these input/output processing routines must be replaced or altered. Graphic application programs may be designed without regard to which particular processing routine will be used. So far as the application program is concerned, device-independence has been achieved.

FIG. 1. STAND-ALONE GRAPHIC CONFIGURATION

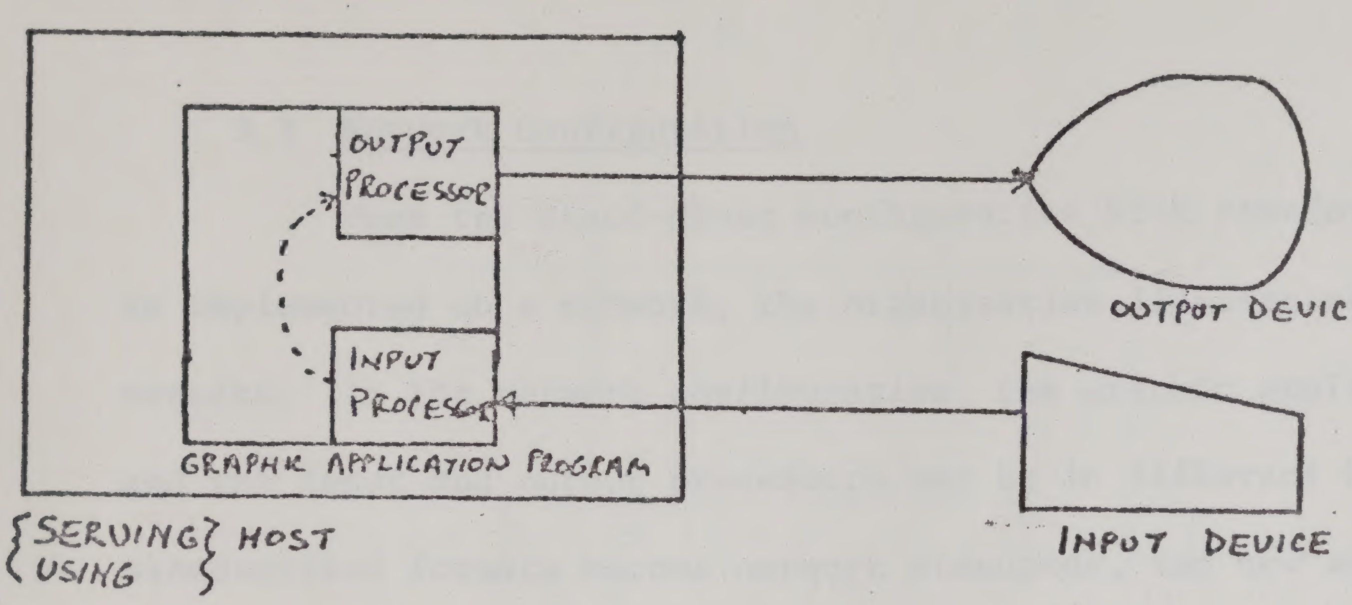
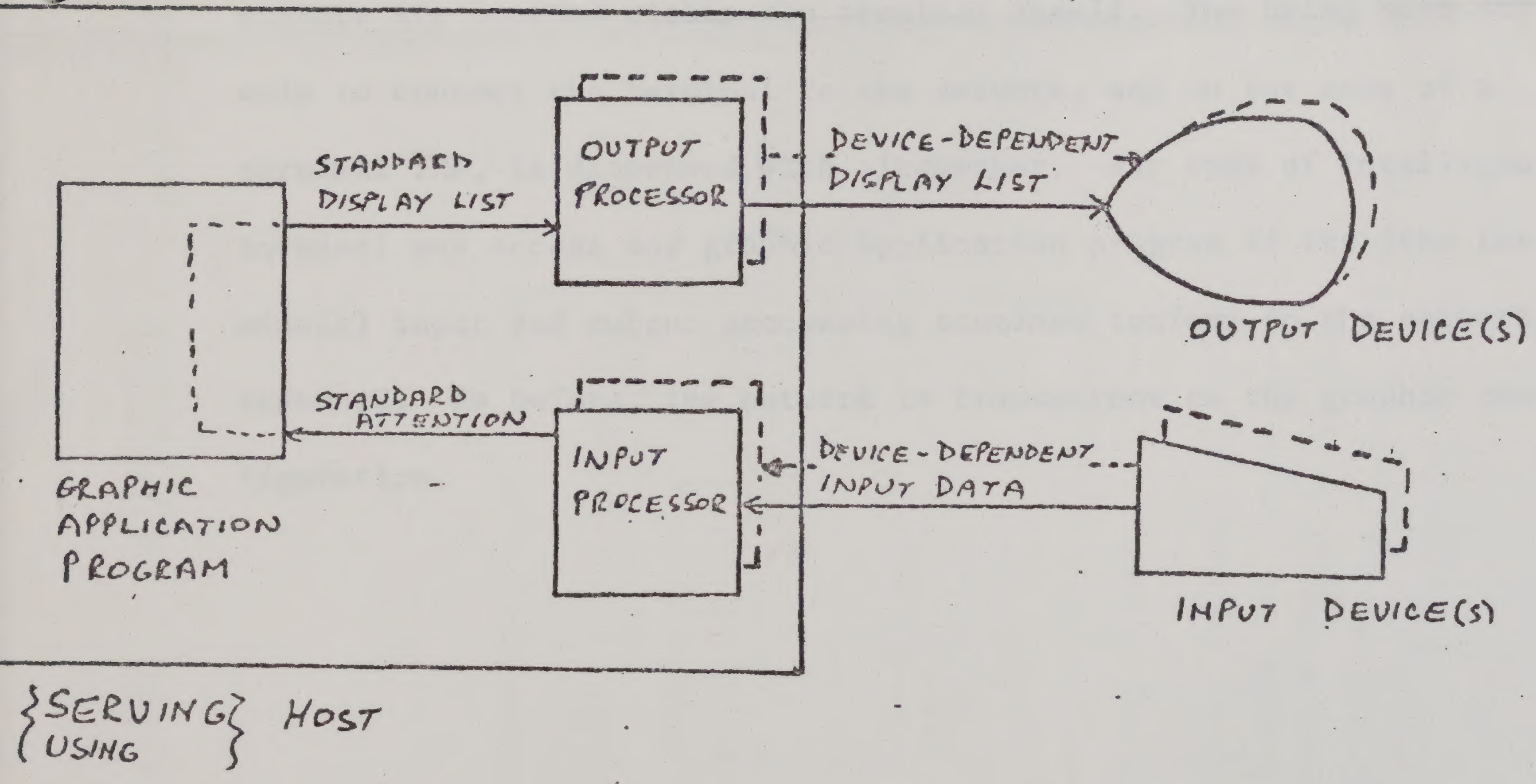

OUTPUT PROCESSOR
INPUT PROCESSOR
GRAPHIC APPLICATION PROGRAM
{SERVING / USING} HOST
OUTPUT DEVICE
INPUT DEVICE

FIG. 2. STAND-ALONE CONFIGURATION WITH STANDARDIZED INPUT AND OUTPUT FORMATS


STANDARD DISPLAY LIST
OUTPUT PROCESSOR
DEVICE-DEPENDENT DISPLAY LIST
OUTPUT DEVICE(S)
STANDARD ATTENTION
INPUT PROCESSOR
DEVICE-DEPENDENT INPUT DATA
INPUT DEVICE(S)
GRAPHIC APPLICATION PROGRAM
{SERVING / USING} HOST

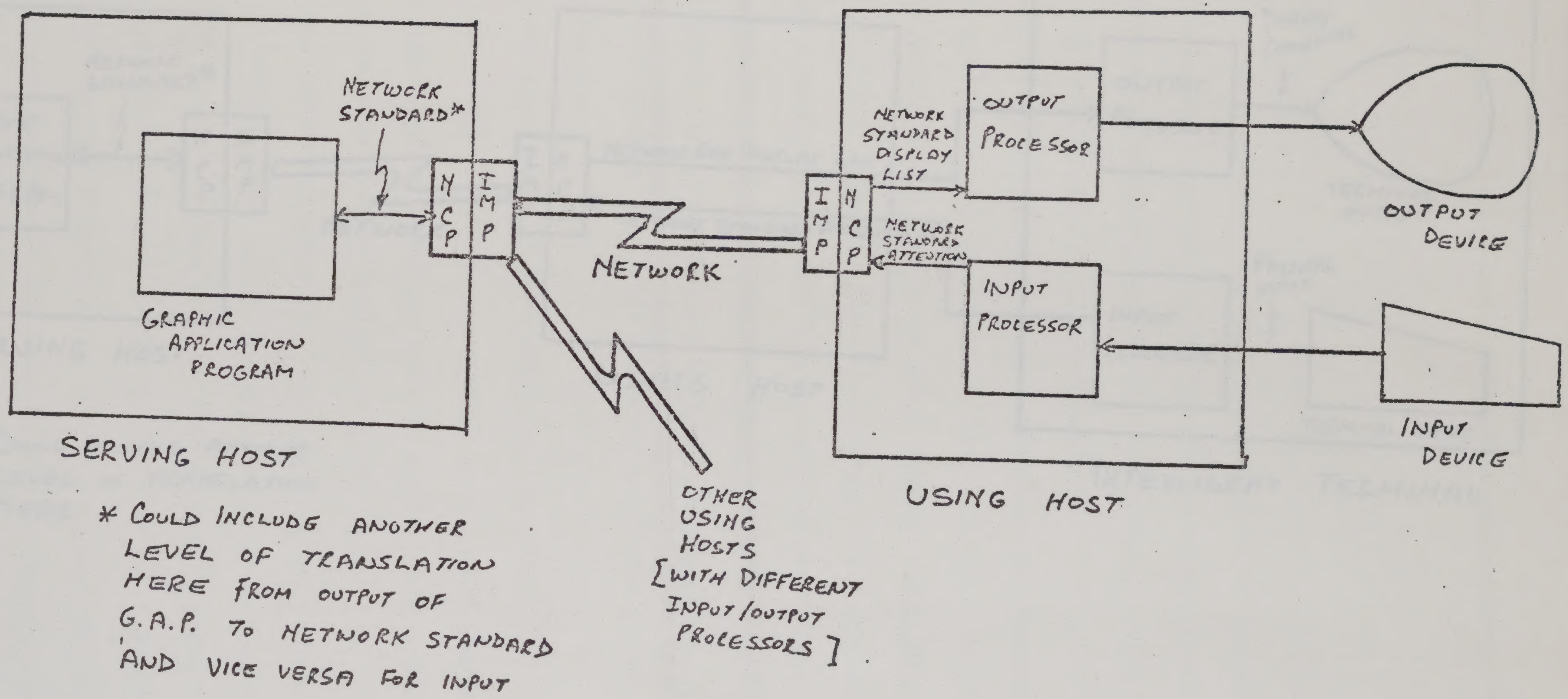
3.3 Network Configuration

When the stand-alone configuration with standardized formats is implemented on a network, the organization illustrated in Figure 3 results. In the network configuration, the graphic application program and the input and output processors may be in different hosts. The standardized formats become network standards, and now any using host with input/output processors conforming to the standard can access the graphic application program in the serving host. The network is transparent to the graphic configuration.

3.4 Network Configuration with Intelligent Terminal

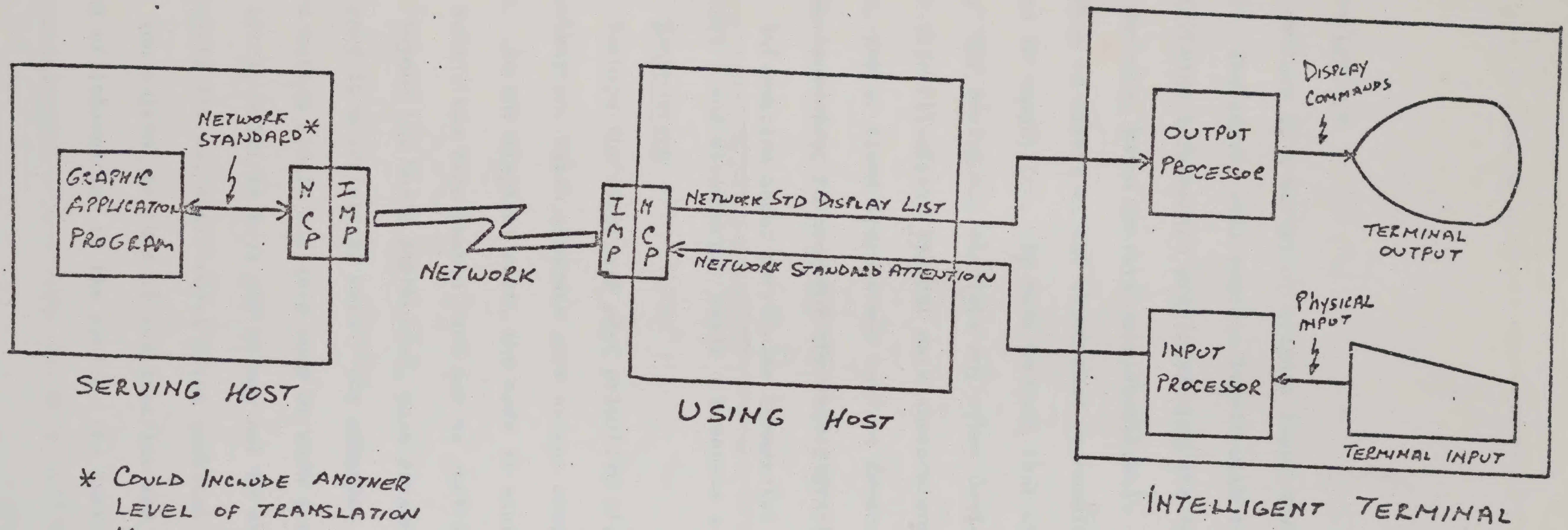
The case of an intelligent graphics terminal configured in the network is illustrated in Figure 4. Here, input and output processors are located within the terminal itself. The using host serves only to connect the terminal to the network, and in the case of a terminal IMP, is dispensed with altogether. Any type of intelligent terminal may access any graphic application program if its (the terminals) input and output processing routines conform to the network standard. As before, the network is transparent to the graphic configuration.

FIG. 3. NETWORK CONFIGURATION


NETWORK STANDARD*
GRAPHIC APPLICATION PROGRAM
NCP
IMP
SERVING HOST
NETWORK
OTHER USING HOSTS [WITH DIFFERENT INPUT/OUTPUT PROCESSORS]
IMP
NCP
NETWORK STANDARD DISPLAY LIST
NETWORK STANDARD ATTENTION
OUTPUT PROCESSOR
INPUT PROCESSOR
USING HOST
OUTPUT DEVICE
INPUT DEVICE

* COULD INCLUDE ANOTHER LEVEL OF TRANSLATION HERE FROM OUTPUT OF G.A.P. TO NETWORK STANDARD AND VICE VERSA FOR INPUT

F 4. NETWORK CONFIGURATION WITH INTELLIGENT TERMINAL


GRAPHIC APPLICATION PROGRAM
NETWORK STANDARD*
NCP
IMP
SERVING HOST
NETWORK
IMP
NCP
NETWORK STD DISPLAY LIST
NETWORK STANDARD ATTENTION
USING HOST
OUTPUT PROCESSOR
DISPLAY COMMANDS
TERMINAL OUTPUT
INPUT PROCESSOR
PHYSICAL INPUT
TERMINAL INPUT
INTELLIGENT TERMINAL

* COULD INCLUDE ANOTHER LEVEL OF TRANSLATION HERE

4.0 INPUT DEVICES

We now turn to a survey of graphic input devices as suggested in RFC 87. The survey will concern itself with the characteristics of the attention information presented when the device is used (rather than, for example, human factors considerations).

We wish to stress at the onset that we consider all devices equivalent in capability. By this we mean that with appropriate programming, any device can simulate any other device. Throughout the survey we will illustrate typical data conversions which might be performed, and at times discuss how various devices may be simulated.

It is convenient to consider the characteristics of classes of devices. Information about particular commercial devices may be found in reference 5 and elsewhere. Table I presents a device class summary.

4.1 Pushbuttons

Perhaps the first and most primitive class of input devices is the pushbutton, which presents some unique code to the system when depressed. In the simplest case, the code is equivalent to the knowledge that the button has been pushed, and may be just a flag.

Beyond the basic pushbutton, more advanced key devices have been designed in a variety of ways. For example, each key may be associated with a single bit in a word or with a complex pattern (character or byte), multiple keys may or may not be able to be struck simultaneously (if so, their codes being combined in some defined way).

The salient feature of the function key is that it presents two pieces of information to the system: the fact that a keystroke

has occurred (which may be implicit), and some unique code related to it.

More elaborate keyboards, be they teletypes or solid state devices with elaborate "overlays", are merely special cases of function keys. They present the same information, attention source plus a unique code. The fact that such a code may be associated with a displayable character is at this stage only incidental.

Since keyboards permit the entry of arbitrary codes, particular sequences of codes may easily be defined to simulate other devices. If local logic permits, codes may be accumulated until a complete sequence is entered and then be reformatted to exactly the same format as the device being simulated would have produced.

Pointing devices such as light pens and tablets may be simulated by associating particular keys with screen directions (up, down, right, left) and using them to position a pointer on the screen face. This is facilitated on terminals with a hardware connection between keys and a cursor symbol.

4.2 Analog Devices

The next most basic class of input devices are those which consist of analog to digital converters. These include simple shaft encoders, mouse and trackball. These devices all produce a digital output proportional to an analog input, in this case, the rotation of a shaft. Several of these inputs may be presented together, as in the case of the mouse and trackball.

has occurred (which may be implicitly) and some unique code related to it.

Those electronic keyboards, as those calculator or solid state devices with elaborate "overlays", are really special cases of function keys. They present the same information, a function key as plus a unique code. The fact that such a case may be associated with a displayable character is of this usage quite irrelevant.

Since keyboards permit the entry of arbitrary codes, particular sequences of codes may easily be defined to simulate other devices. If enough logic permits, codes may be accumulated until a complete sequence is entered and then be implemented to exactly the same format as the device being simulated would have generated.

Pointing devices such as light pens and tablets may be simulated by associating particular keys with screen directions (up, down, right, left) and using them to position a pointer on the screen. This is facilitated on terminals with a hardware connection between keys and a cursor symbol.

2.5 Analog Devices

The next most basic class of input devices are those which consist of analog to digital converters. These include simple shaft encoders, knobs and joysticks. These devices all produce a digital output proportional to an analog input, in this case, the rotation of a shaft. Several of these inputs may be presented together, as in the case of the mouse and trackball.

These devices all present as input a device identification (which may be implicit depending on the hardware method of input) together with a number of digital codes from the same number of analog devices. The length of the code is arbitrary and may or may not relate to such measures as the maximum raster count of the display screen.

Analog devices are often used as pointing devices by using the input to control the movement of a cursor on the screen face. This method is superior to the use of a keyboard, since very smooth and rapid movement may be obtained.

4.3 Tablets

A tablet consists of a flat surface on which (X,Y) position may be indicated with a stylus. If position changes can be registered rapidly enough, arbitrary curves may be digitized by tracing them.

There are a variety of devices utilizing a variety of techniques comprising this class. Included are such diverse techniques as variable resistance, variable capacitance, and ultrasonics, to mention a few of the devices on the market. The surface may be horizontal or vertical and may even be superimposed on the screen itself. A variety of styli have been used, including the operator's finger. A device (the Lincoln Wand) has also been demonstrated which may be manipulated in space to yield a position in three dimensions (X, Y, Z).

These devices all present a device identification (which may be implicit), and a position value, which is most often a coordinate pair, but which may be a triple.

4.4 Light Pen

Light pens are devices which relate the occurrence of an attention to the time in the refresh cycle when a particular point is illuminated on the screen. The display generators are generally stopped when the attention occurs, to permit either the display list "P" register or the (X, Y) beam position registers, or both to be presented as attention data. Often times this is not enough, as what is desired is some value which serves to identify the image which the pen detected. In such cases local hardware and/or software is utilized to obtain this information, which may be as simple as a single identification code or as elaborate as a genealogical push down list.

Light pens present as input a device identification (which may be implicit) and at least one of the following: memory address, (X, Y) position, item identification.

Light pens may be used to simulate keyboards by displaying "light buttons" on the screen associated with particular physical buttons. Detecting on a light button is logically equivalent to pushing the related key.

4.5 Internal Attentions

Internal attentions are stimuli arising not from operator action, but from various sources within the terminal such as a screen edge violation (overflow) or a programmed trap. Such attentions present information in much the same way as the operator input devices already discussed. This information consists of an attention source identification (equivalent to device identification, and which may

again, be implicit) and appropriate data, which, for the two examples given, will generally be a memory address.

Programmed traps are often used to permit mode changes (e.g., enable or disable light pen operation) during the execution of the display list. Edge violation might occur when an image is being relocated in response to operator input. We must provide for describing such attentions, since then cannot always be handled locally in the terminal.

4.6 Logical Attentions

We may generalize the concept of an attention from a stimulus from a physical source to a logically generated stimulus resulting from some program condition which may or may not cause an interrupt. (Programmed traps were classified as internal attentions because, by definition, they cause an interrupt or set a hardware flag). Logical attentions are generally "input" by setting a software flag which some control program can periodically inspect. For example, logical attentions may be designed to detect when a software-defined edge violation occurs (of a region less than full screen) or when a light button is picked. There is no general form for the information generated by logical attentions, since they are programmable, rather than hardware-bound. The best we can do is to say they consist of an identification and appropriate data. Actually, logical attentions most often simulate physical attentions, and so each may be placed in one of the classes already described.

TABLE I

INPUT DEVICE SUMMARY

Device Class	Device Examples	Typical Output
Button	Teletype	1 Character
	Function Key with Overlay	1 Character & overlay code
	Buffered Keyboard	n Characters
A/D Converter	Shaft Encoder	$\Delta \alpha$
	Mouse	$(\Delta \alpha, \Delta \beta)$
Tablet	Rand Tables &	(X,Y)
	Lincoln Word	(X,Y,Z)
Light Pen	Light Pen	P (memory address)
	Light Pen	(X,Y)
	Light Pen & Local Software	Item name
	Light Pen & Local Software	Item name stack
Internal	Program Trap	P (memory address)
	Screen Overflow	P (memory address)
Logical Attention	Any of the above	Any of the above

5.0 INTELLIGENT TERMINALS

As has been indicated, the question of what data results from which inputs is complicated when "intelligent terminals" are considered. An intelligent terminal has the ability to modify the data presented by the input device hardware. In a sense then, all of the outputs of an intelligent terminal may be considered as logical attentions. The logical complexity of such attentions may be very great indeed. For example, such a terminal might be programmed to perform sketching functions, so that the net effect of a keystroke and a light pen hit might be the deletion of a portion of the picture together with some coded message to the effect. A technique has even been developed which permits the light pen operator to simulate the use of a shaft encoder by twisting his wrist which holding the pen over a tracking symbol(7).

Some intelligent terminal systems have been developed which permit the terminal operator to modify the picture and the local data structure independently.(2) Thus, the need for a very expressive protocol from terminal to central computer becomes apparent, so that notice of such local processing may be communicated to the central program.

6.0 NETWORK PROTOCOL GUIDELINES

We now suggest a format for a (third level) network protocol from terminal to serving host which is sufficiently open-ended to permit any type of attention to be communicated. It is not the intent here to formally propose such a protocol down to the level of "this bit means that." When such details are decided, a Network Standard Attention will have been defined.

The attention protocol has three basic elements: device identification, data identification, and data.

6.1 Device Identification

The device identification field must be sufficiently large to permit the unique identification of any type of device in the network. If two or more identifical input devices exist at different nodes in the network, it is not necessary to distinguish among them in this field. However, if a keyborad, for example, has keys which are logically different, such as typewriter keys and function keys, the distinction should be made in the identification field, rather than requiring an analysis of the data. Further, if two different devices are logically equivalent, as a physical keyboard and light buttons, they may be so treated by not having identification codes different from each other.

Somewhere in the network (and possibly at each host supporting a graphic application) a table should be kept of the input device types and their characteristics. It may be convenient to

organize the device identification field so that a subfield identifies the device class, as discussed previously.

6.2 Data Identification

The device identification field is intended to contain a description of the data field which follows. Information which might be provided here includes number of units (bits, words, bytes) of data which follow, qualitative description of the data (character code, memory address, cartesian coordinates, item name, etc.), and data format information. It may be desirable, for the sake of uniformity, to include this information even when it is somewhat redundant.

6.3 Data

Lastly comes the data itself (perhaps an anticlimax at this point !) which, as should be clear by now, may be of arbitrary length and organization.

Bibliography

1. Cotton, I. "Languages for Graphic Attention-Handling." Proc. Computer Graphics 70 Symposium, Brunel University, 1970.

2. Cotton, I. & F. Greatorex "Data Structures & Techniques for Remote Computer Graphics," Proc. FJCC, 1968, pp. 533-544.

3. Crocker, S. "Proposal for a Network Standard Format for a Data Stream to Control Graphics Display." ARPA Network Working Group, RFC #86, 1971.

4. Harslem, E. & J. Heafner "Some Thoughts on Network Graphics," ARPA Network Working Group, RFC #94, 1971.

5. Keast, D. "Survey of Graphic Input Devices," *Machine Design*. August 3, 1967, pp. 114-120.

6. McConnell, J. "Response to RFC #86," ARPA Network Working Group, RFC #125, 1971.

7. Newman, W. "A Graphical Technique for Numerical Input," *Computer J.*, May 1968, pp. 63-64.

8. Vezza, A. "Topic for Discussion at the Next Network Working Group Meeting." ARPA Network Working Group, RFC #87, 1971.